**TANIA MAINA.**

**SCT212-0179/2021.**

**BCT2408: COMPUTER ARCHTECTURE.**

E1

a)

**Pipeline Stages:**

* IF – Instruction Fetch
* ID – Instruction Decode / Register Fetch
* EX – Execute / Address Calculation
* MEM – Memory Access
* WB – Write Back

**Assumptions:**

**5-stage pipeline:** Instruction Fetch (IF) → Instruction Decode (ID)→ Execute (EX) → Memory (MEM) → Write Back (WB)

* No forwarding or bypassing
* Registers can be read and written in the same clock cycle — so data written in WB in cycle n can be read in ID of cycle n+1
* Branch resolved in ID stage and is followed by a pipeline flush (stall)
* Memory accesses take 1 cycle
* Separate instruction and data memory
* Initial R3 = R2 + 396 → 396 / 4 = 99 loop iterations

Dependencies (No Forwarding)

**1. LD → DADDI (R1)**

LD writes in WB (cycle 5)

DADDI reads in ID (needs result in cycle 3)

2 stalls are needed

**2. DADDI → SD (R1)**

DADDI writes in WB (cycle 7), SD reads in ID (would be cycle 5)

Need to wait until DADDI completes → 1 stall

**3. DADDI R2 → DSUB (R2)**

DADDI writes in WB (cycle 9), DSUB reads in ID (cycle 7)

Need 2 stalls

**4. DSUB → BNEZ (R4)**

DSUB writes in WB (cycle 11), BNEZ reads in ID (cycle 9)

Need 2 stalls

**5. Branch flush after BNEZ**

BNEZ resolves in ID → if branch taken, 2 instructions fetched after BNEZ are flushed

**Pipeline representation diagram.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Cycle** | **LD** | **DADDI** | **SD** | **DADDI2** | **DSUB** | **BNEZ** |  |
| 1 | IF |  |  |  |  |  |  |
| 2 | ID |  |  |  |  |  |  |
| 33 | EX | STALL |  |  |  |  | DADDI stalled (waiting for R1) |
| 4 | MEM | STALL |  |  |  |  |  |
| 5 | WB | IF |  |  |  |  | DADDI issued now that R1 is ready |
| 6 |  | ID |  |  |  |  |  |
| 7 |  | EX | STALL |  |  |  | SD stalled (waiting for R1) |
| 8 |  | MEM | IF |  |  |  |  |
| 9 |  | WB | ID |  |  |  |  |
| 10 |  |  | EX | STALL |  |  | DADDI2 stalled (waiting for R2) |
| 11 |  |  | MEM | STALL |  |  |  |
| 12 |  |  | WB | IF |  |  |  |
| 13 |  |  |  | ID |  |  |  |
| 14 |  |  |  | EX | STALL |  | DSUB stalled (waiting for R2) |
| 15 |  |  |  | MEM | STALL |  |  |
| 16 |  |  |  | WB | IF |  |  |
| 17 |  |  |  |  | ID |  |  |
| 18 |  |  |  |  | EX | STALL | BNEZ stalled (waiting for R4) |
| 19 |  |  |  |  | MEM | STALL |  |
| 20 |  |  |  |  | WB | IF | BNEZ issued now |
| 21 |  |  |  |  |  | ID | Branch resolved → pipeline flushes |
| 22 |  |  |  |  |  |  | FLUSH 1 |
| 23 |  |  |  |  |  |  | FLUSH 2 |
| 24 | IF (next loop) |  |  |  |  | Start of next iteration |  |

Each iteration takes 24 cycles (due to stalls and flush)

**Total cycle count for all iterations:**

98 iterations: 98 × 24 = 2,352 cycles

Final iteration: BNEZ fails → loop ends, no flush, only 22 cycles needed

Therefore; Total = 2,352 + 22 = 2,374 cycles

b)

**Assumptions:**

* Forwarding/bypassing is available – so most data hazards are resolved with minimal or no stalls.
* Branches are predicted not taken, but resolved in ID stage
* Separate instruction and data memory – no structural hazard.
* Memory access takes 1 cycle.
* Each loop adds 4 to R2, R3 = R2 + 396 → 99 iterations.

**Dependencies:**

**With Forwarding:**

LD → DADDI: RAW hazard on R1 → 1 stall needed because LD's data is not available until MEM stage (cycle 4), while DADDI needs it in EX (cycle 3).  
 1 stall needed even with forwarding (load-use hazard).

DADDI → SD: Forward from DADDI’s EX or MEM to SD’s EX → no stall.

DADDI R2 → DSUB: Forward from DADDI's EX/MEM → no stall.

DSUB → BNEZ: Forward R4 from DSUB’s EX → no stall.

**Pipeline representation diagram.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Cycle** | **LD** | **DADDI** | **SD** | **DADDI2** | **DSUB** | **BNEZ** |
| 1 | IF |  |  |  |  |  |
| 2 | ID | IF |  |  |  |  |
| 3 | EX | **STALL** | IF |  |  |  |
| 4 | MEM | ID | ID | IF |  |  |
| 5 | WB | EX | EX | ID | IF |  |
| 6 |  | MEM | MEM | EX | ID | IF |
| 7 |  | WB | WB | MEM | EX | ID |
| 8 |  |  |  | WB | MEM | EX |
| 9 |  |  |  |  | WB | MEM |
| 10 |  |  |  |  |  | WB |

Total = 10 cycles per loop iteration

**Loop iterations:**

Since branch is predicted not taken, no stall if prediction is correct.

But on final iteration, branch will be mispredicted → 2-cycle penalty.

**Total iterations:**

R2 starts at x

R3 = R2 + 396 → every iteration increases R2 by 4

So total = 396 / 4 = 99 iterations

**Total Cycles:**

98 iterations with no misprediction = 98 × 10 = 980 cycles

99th iteration (misprediction) = 10 + 2 = 12 cycles

Total = 980 + 12 = 992 cycles

Only 1 stall per iteration (load-use), and 2-cycle penalty for final branch misprediction

c)

**Assumptions:**

* Classic 5-stage RISC pipeline: IF → ID → EX → MEM → WB
* Forwarding is available
* Single-cycle delayed branch: the instruction after a branch always executes, regardless of whether the branch is taken or not.
* Separate instruction & data memory: no structural hazard.
* Memory access = 1 cycle
* Initial R3 = R2 + 396 ⇒ 99 iterations

**Instruction reordering to fill the branch delay slot**

Move DADDI R2, R2, 4 (increment pointer) into the delay slot because it is always needed before the next loop iteration and move DSUB before BNEZ.

**Optimized instruction schedule:**

loop: LD R1, 0(R2)  
 DADDI R1, R1, 1  
 SD 0(R2), R1  
 DSUB R4, R3, R2  
 BNEZ R4, loop  
 DADDI R2, R2, 4; branch delay slot

**Data hazards (with forwarding)**

LD → DADDI (R1): 1 cycle stall (load-use hazard)

Other instructions benefit from forwarding:

DADDI (R1) → SD: forwarded

DSUB → BNEZ: forwarded

R2 is read by DSUB and written by DADDI in the delay slot → OK, because R2 is used before it’s updated.

**Pipeline representation diagram**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Cycle** | **LD** | **DADDI** | **SD** | **DSUB** | **BNEZ** | **DADDI (R2)** |
| 1 | IF |  |  |  |  |  |
| 2 | ID | IF |  |  |  |  |
| 3 | EX | **STALL** | IF |  |  |  |
| 4 | MEM | ID | ID | IF |  |  |
| 5 | WB | EX | EX | ID | IF |  |
| 6 |  | MEM | MEM | EX | ID | IF |
| 7 |  | WB | WB | MEM | EX | ID |
| 8 |  |  |  | WB | MEM | EX |
| 9 |  |  |  |  | WB | MEM |
| 10 |  |  |  |  |  | WB |

Total per iteration: 10 cycles

**Total loop cycles:**

99 iterations, no branch misprediction penalty (delay slot is correctly used):

99 iterations × 10 cycles = 990 cycles